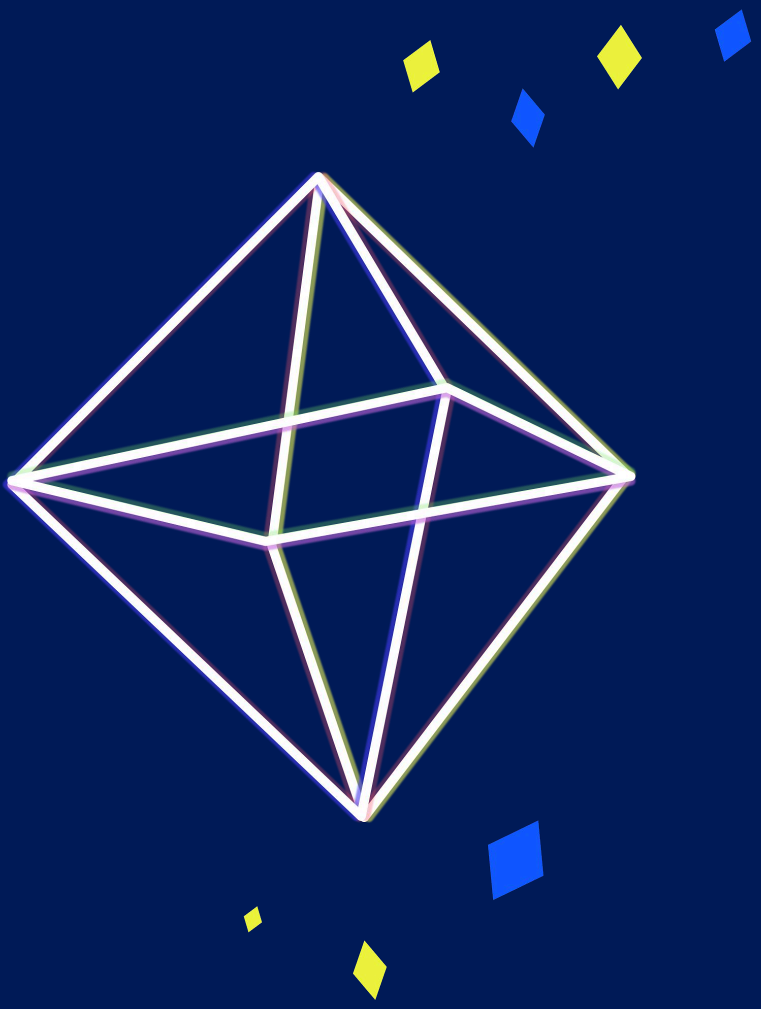
THE STATE OF

Software Engineering Excellence 2025



harness

harness

The engineering excellence gap is real. And it's costing you millions.

CrowdStrike's global outage. SolarWinds' supply chain breach. The Colonial Pipeline ransomware attack. These aren't isolated incidents—they're symptoms of a systemic crisis in how we build, deploy, and secure software. While headlines scream about AI transforming everything, the uncomfortable truth is that most engineering teams are drowning in technical debt, manual processes, and security vulnerabilities that should have been solved a decade ago.

But here's what's fascinating: the organizations getting this right aren't just avoiding disasters—they're unlocking millions in hidden value. They're shipping features 60% faster, cutting cloud costs by 15%, and reducing security incidents to near zero. The difference isn't talent or budget. It's knowing exactly where the gaps are and how to close them systematically.

This report exposes the reality of engineering maturity in 2025, based on candid responses from over 650 engineering leaders who completed the Engineering Excellence Maturity Assessment, developed by EngineeringX, a community of hundreds of CTOs and Engineering VPs. What we found will surprise you: even organizations that think they're "DevOps mature" are missing fundamental practices that separate industry leaders from everyone else.

The five dimensions we measured: Developer Experience, DevOps Modernization, Optimization, Quality & Resilience, and Secure Software Development, reveal where the biggest opportunities lie. By examining this comprehensive data, this report aims to illuminate the adoption of engineering excellence best practices, identify common challenges, and highlight actionable areas for improvement as organizations continue their vital journey toward engineering excellence.

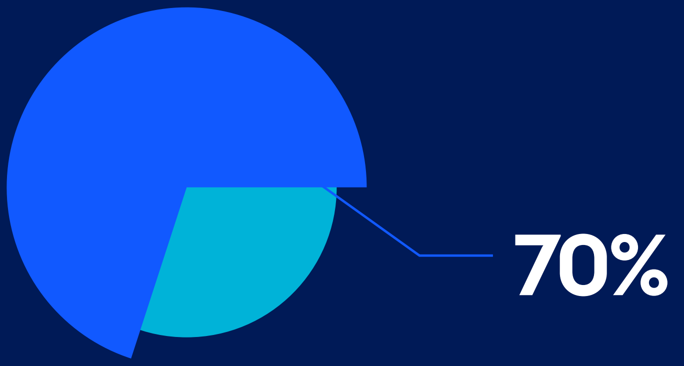
01: Developer Experience

Developer experience is a cornerstone of success for today's digital-first organizations and is essential to keeping developers happy and productive. Establishing a mature engineering practice hinges significantly on cultivating an exceptional developer experience by arming developers with the tools, capabilities, and resources they need to work quickly and efficiently. This involves optimizing every touchpoint in an engineer's workflow, from rapidly provisioning environments and providing readily discoverable documentation to streamlining planning processes and ensuring efficient code quality practices.

Planning and requirements process

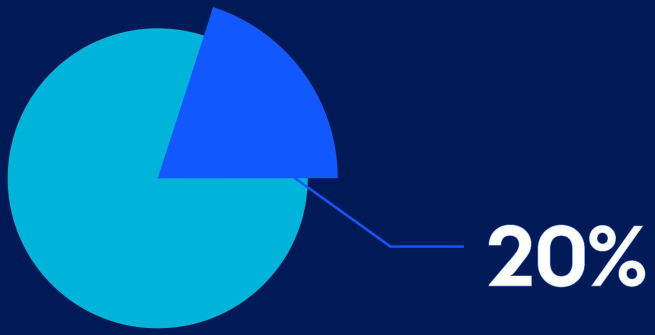
Effective planning and clear requirements are foundational to a positive developer experience. Engineers face frustration, rework, and a reduced sense of accomplishment when requirements are ambiguous or constantly shifting. A well-defined planning process, including precise acceptance criteria, minimizes friction, boosts confidence, and allows development teams to focus on building value rather than deciphering ill-defined goals.

However, many organizations struggle with this:



1 in 4 engineering leaders say

more than 70% of requirements do not have clearly defined acceptance criteria.



Over half (54%) of engineering leaders

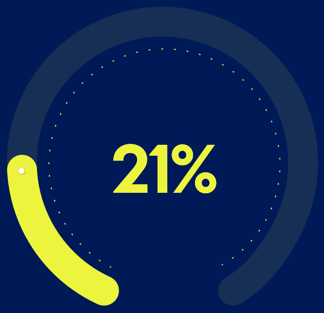
have seen an average scope creep of above 20% in the last 12 sprint cycles.

To mature your planning and requirements process, prioritize defining clear and consistent acceptance criteria for all features. Implement tools and practices that foster communication between product and engineering, reducing ambiguity and ensuring predictable scope.

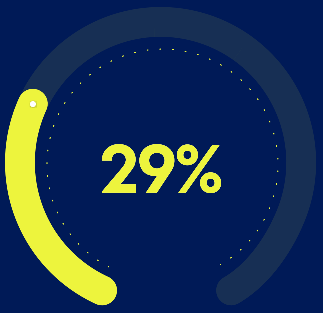
Discoverability and documentation

Engineering teams need instant access to accurate information about the software and services they're working on. That enables them to make more informed decisions and follow a consistent development methodology. However, they are also under pressure to deliver quickly, so engineering leaders must ensure these processes don't create extra effort for developers. A comprehensive software catalog, effectively documenting metadata, ownership, and service status, is crucial for reducing friction and enabling engineers to focus on innovation.

This is a significant challenge across the industry:



Only 21% of engineering teams have a software catalog that is automatically updated with changes.



Nearly a third (29%) of teams have no software catalog at all.

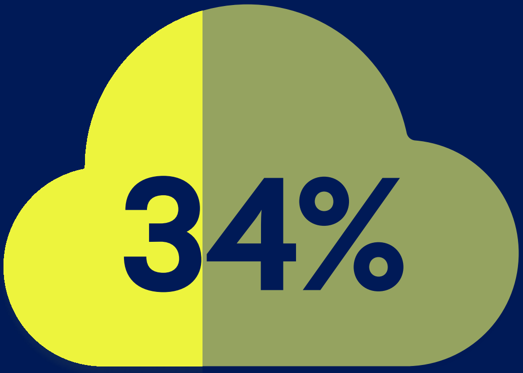
To mature discoverability and documentation, implement an automated software catalog, ideally via an IDP. This centralizes metadata, ownership, and status, ensuring real-time accuracy without manual developer effort.

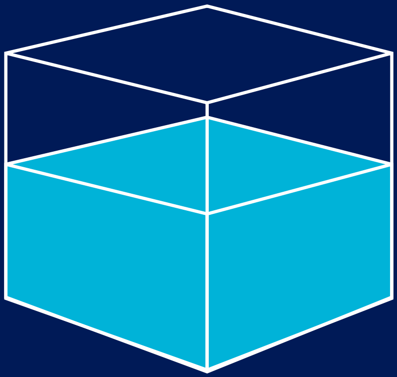
Developer environments

Organizations striving for optimal software delivery recognize the critical role of efficient developer environments. Delays in provisioning or inconsistencies across these setups directly impede productivity, slow iteration cycles, and introduce unnecessary friction into the development workflow. Streamlining how teams access and utilize pre-built, standardized environments is paramount for accelerating feature delivery and maintaining developer velocity.

Despite its value, challenges in environment provisioning persist across the industry:

Only a third (34%) of engineering teams can quickly spin up pre-built developer environments in the cloud.





67% of engineering leaders say their developers **can NOT** build and test dev environments within 15 min.

To mature your development process, focus on providing developers with intuitive, self-service tools for rapid environment provisioning and consistent configurations. This frees them from manual setup overhead and lets them focus on value-added tasks. An effective approach is incorporating automated environment provisioning into Internal Developer Portals (IDPs).

Development process & hygiene

Robust development processes and stringent coding hygiene are fundamental to a productive and satisfying developer experience. When code reviews become bottlenecks and commit sizes grow unwieldy, the entire development cycle slows, impacting efficiency and the ability to maintain high-quality software. Addressing these foundational elements is crucial for accelerating innovation and ensuring reliable releases.

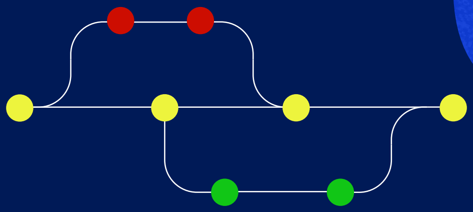
Industry data reveals common challenges in this area:

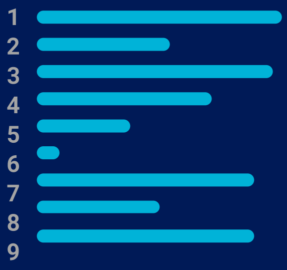
61%

Almost two-thirds (61%)

of engineering leaders say code reviews take over a day in their organization.

35% of engineering teams do not consistently follow the branching strategy for all QA, dev, and infrastructure repos





14% of engineering leaders say their average code commit is less than 30 lines of code

To mature development processes and hygiene, prioritize efficient code review workflows and encourage smaller, more frequent code commits. Implement automated checks and clear branching strategies to maintain code quality and accelerate integration without burdening developers.

Learning and development

To deliver a truly great developer experience, engineering leaders need to ensure there are opportunities for personal development and learning new skills. Investing in continuous learning keeps teams at the forefront of technology, prevents skill stagnation, and significantly boosts morale and retention.

However, many organizations face a gap in this area:

19% of engineering leaders say they have a structured curriculum for upskilling and reskilling engineers.

To mature learning and development initiatives, leaders should focus on establishing clear curricula and prescriptive learning paths. These programs, including internal training and external certifications, are crucial for effective upskilling and reskilling, ensuring engineers consistently grow their capabilities.

The Bottom Line

Streamlining the developer experience through better documentation, planning, and environment provisioning accelerates new developer onboarding. Equipping engineers with efficient tools and workflows cuts the time new hires take to reach full productivity. For every 1000 developers, organizations have an opportunity to realize the following savings:

Formula:

1k Devs x $100k Annual Salary = **$100M**

1k Devs x 2000 hours = **2M** annual work hours

15% growth & attrition backfill = 150 developers onboarded x 8 weeks onboarding per dev = **48k** onboarding hours per year

48k hrs of onboarding x $50 per hr = **$2.4M** per year in lost productivity due to lengthy onboarding

40-60% estimated faster onboarding from DevEx best practices

$1M-$1.4M annual savings **JUST** in onboarding costs

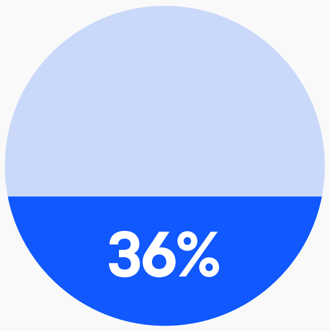
02: DevOps Modernization

DevOps modernization is not merely about adopting new tools; it's a strategic imperative to transform how software is built, delivered, and secured, laying the groundwork for a truly mature engineering practice. It encompasses streamlining critical processes—from build pipelines and deployment automation to advanced strategies and robust security—to drive efficiency, reduce risk, and accelerate the pace of innovation. Organizations can significantly improve their operational stability and developer productivity by addressing these foundational areas, thus establishing a mature engineering practice.

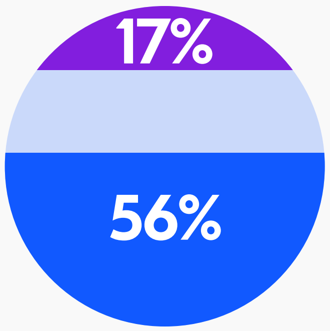
Build process

Standardized build processes are essential to maintain software quality and security while enabling developers to accelerate delivery through modern DevOps practices. These pipelines should include automated quality gates to prevent bad code from reaching production and reduce context switching for developers, ultimately contributing to a more efficient and reliable software delivery lifecycle.

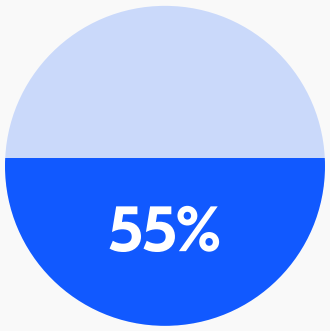
Despite this importance, many organizations face significant hurdles:



Over a third (36%) of organizations do not have standardized build pipelines.



While 56% of organizations execute all their test cases as part of the build pipeline, only 17% have test cases selected intelligently based on applicable code changes.



55% of build pipelines are not gated or have gates that are not enforced.

To mature your build process, focus on establishing standardized, templated build pipelines that incorporate automated quality gates. These gates should intelligently run relevant tests and enforce policies to prevent non-compliant or faulty code from progressing, thereby improving quality and developer efficiency.

Deployment process

The deployment process is the critical nexus where development meets operations, fundamentally defining an organization's velocity and reliability in the journey towards DevOps modernization. An optimized deployment pipeline is essential for delivering software rapidly and consistently, ensuring that new features and fixes reach production safely and efficiently. This section explores various facets of modern deployment, from automation and strategies to rollbacks and security.

Deployment automation

Standardized and automated deployment processes are cornerstones of effective DevOps modernization, directly optimizing the delivery cycle. When manual steps are minimized, organizations can accelerate releases, reduce human error, and free up valuable engineering time for innovation. Automating deployment steps allows for consistent, repeatable, and rapid software delivery, which is crucial for modern development velocity.

Despite this clear advantage, significant manual intervention persists:

Nearly 1 in 5 engineering teams do not have standardized or templatized deployment pipelines

50% of application code deployments still rely on manual steps.

64% of infrastructure code deployments still rely on manual steps.

69% of database changes still rely on manual deployment steps.

To mature deployment automation, focus on standardizing and templating all deployment pipelines. Prioritize automating every possible step—from code and infrastructure to database changes—to eliminate manual intervention and achieve consistent, rapid, and reliable releases.

Deployment strategies

Modern deployment strategies such as rolling, canary, and blue/green updates, GitOps, and feature flagging are foundational to engineering maturity. Implementing these practices enables organizations to control release risk, perform phased rollouts, and ensure stability in production, all of which are crucial for effective DevOps modernization. Yet many organizations have not yet incorporated these practices into their DevOps workflows.

The adoption of these strategies remains a challenge:

36%

of engineering teams don’t use deployment strategies like rolling update, canary, blue/green, and GitOps.

76%

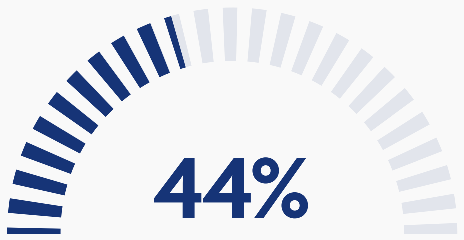
of engineering teams don’t use feature flags or don’t have a process for rolling them out and managing their lifecycle.

To mature deployment strategies, systematically adopt these modern, progressive delivery techniques. Establish clear processes for their implementation and management to reduce deployment risk, accelerate feedback loops, and decouple feature releases from code deployments.

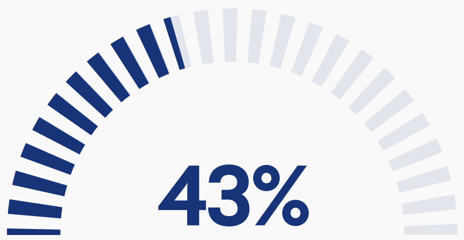
Automated rollbacks

Many organizations manage rollbacks manually or rely on subjective evidence following deployment failures. These methods introduce delays and variability into the recovery process, directly impacting Mean Time To Recovery (MTTR) and hindering overall DevOps modernization efforts. Implementing automated rollback capabilities offers a more consistent and efficient approach to mitigating the impact of failed deployments, enabling swifter resolution and improved operational stability.

The challenges are clear:



of organizations rely on manual rollbacks for failed deployments.



of engineering teams base rollback decisions on subjective evidence rather than data.

To mature your rollback process, prioritize implementing fully automated rollback capabilities for all critical deployments. Supplement automation with data-driven decision-making frameworks, leveraging objective metrics to trigger and validate rollbacks, thus ensuring rapid and reliable recovery.

Deployment security

Engineering leaders must also ensure their deployment processes adhere to security best practices and safeguard sensitive customer and enterprise data from exposure. Modern DevOps includes rigorously controlling access to secrets and personally identifiable information (PII) throughout the software delivery pipeline, preventing vulnerabilities that could lead to breaches or compliance failures.

However, many teams fall short in this critical area:

43% of engineering teams don't store secrets in a centralized secrets manager/HSM (a best practice).

1 in 10 engineering teams store secrets and PII in plain text in their deployment system.

To mature deployment security, prioritize adopting a centralized secrets management solution, such as a Hardware Security Module (HSM) or dedicated secrets manager, across all deployment stages. Enforce policies to prevent sensitive data from being stored in plain text and integrate security checks into pipelines to maintain compliance and protect data integrity.

The Bottom Line

Modernizing DevOps streamlines build and deployment processes, significantly reducing manual steps and freeing up valuable engineering time. With 23% of developer time currently spent on manual build and deployment activities, addressing this manual intervention unlocks substantial efficiencies. For every 1000 developers, organizations have an opportunity to realize the following savings:

Formula:

1k Devs x $100k Annual Salary = **$100M**

1k Devs x 2000 hours = **2M** annual work hours

2 million annual working hours x **23%** spent on manual build & deployment activities = **460k** hours per year

460k hrs x $50 per hr = **$23M** lost annually to build and deployment toil

25-40% reduction in build & deployment toil (conservative estimate)

$6M-$9M annual savings

03: Optimization

Establishing a truly mature engineering practice requires relentless optimization of resource utilization, encompassing systems, costs, and the invaluable effort of your developers. This strategic focus ensures that every investment in cloud infrastructure or engineering hours yields maximum efficiency and value. Organizations can eliminate waste, enhance performance, and drive sustainable growth by continually refining how resources are consumed.

Cloud costs

Managing cloud costs is critical to effective optimization for mature engineering practices. As cloud adoption scales, gaining granular visibility into spending and automating cost reduction efforts becomes essential to maximizing resource utilization, controlling budgets, and ensuring financial efficiency across the organization.

However, many organizations struggle with this fundamental aspect of optimization:

Only 2 of 5

engineering leaders say cloud cost is tracked by their organization's projects, teams, and departments.

12%

of engineering leaders say efforts to reduce cloud costs are automated.

To mature cloud cost optimization, implement automated tools that provide granular, real-time cost visibility down to projects, teams, and workloads. Prioritize automating cost reduction efforts through intelligent resource management and continuous optimization recommendations.

Metrics and insights

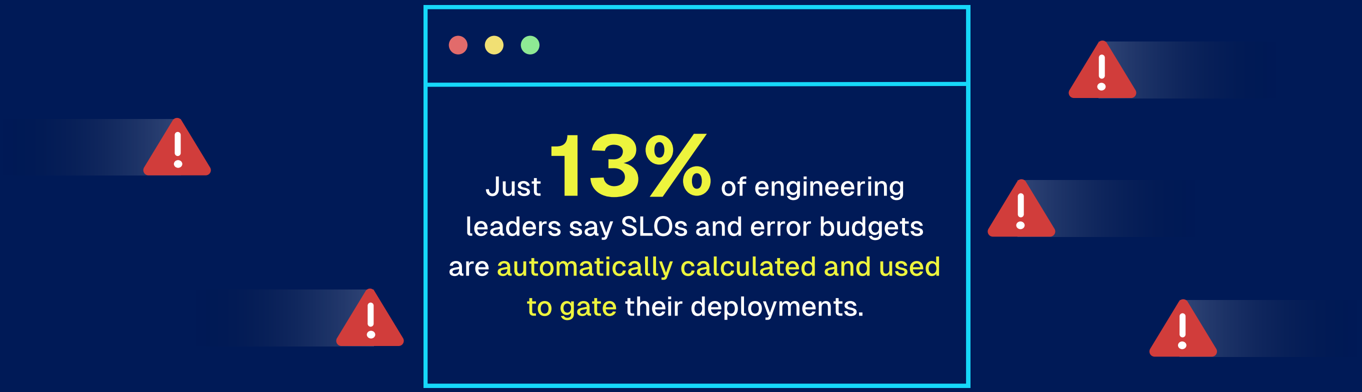
Achieving true optimization in engineering requires a robust approach to metrics and insights. Without a clear understanding of performance data, engineering leaders cannot identify bottlenecks, measure the impact of changes, or make informed decisions to enhance efficiency and resource utilization. Establishing a culture of data-driven analysis is fundamental to maturing DevOps practices and driving continuous improvement.

However, many organizations are still maturing in this area:

17%

of engineering leaders say metrics are analyzed by their team daily or weekly.

Nearly half of engineering leaders say SLOs and error budgets are not defined or calculated for any of their services.



Just **13%** of engineering leaders say SLOs and error budgets are **automatically calculated and used to gate** their deployments.

To mature your metrics and insights, establish clear, actionable key performance indicators (KPIs) for your software delivery lifecycle. Implement automated tools for continuous data collection and analysis, and integrate this intelligence to drive decisions, define SLOs, and gate deployments for consistent performance optimization.

The Bottom Line

Optimizing cloud costs, with granular visibility and automated reduction efforts, unlocks significant financial benefits. This calculationv illustrates the substantial savings organizations can achieve by adopting cloud optimization best practices, aligning with the 10-15% reduction in cloud spend seen by leading FinOps organizations. Organizations that use the cloud have an opportunity to realize the following savings:

Formula:

$35M total cloud spend - $5M cloud marketplace spend = **$30M** optimizable cloud spend

10-15% savings by adopting cloud optimization best practices

$3M-$4.5M reduction in cloud spend

04: Quality & Resilience

At the heart of a mature engineering practice lies the unwavering commitment to software quality and operational resilience. This means rigorously testing applications, proactively building systems that withstand extreme conditions, and establishing robust processes to manage and learn from every incident. By prioritizing these foundational elements, organizations can ensure their software is not only reliable but also capable of enduring real-world challenges, ultimately building trust and driving business continuity.

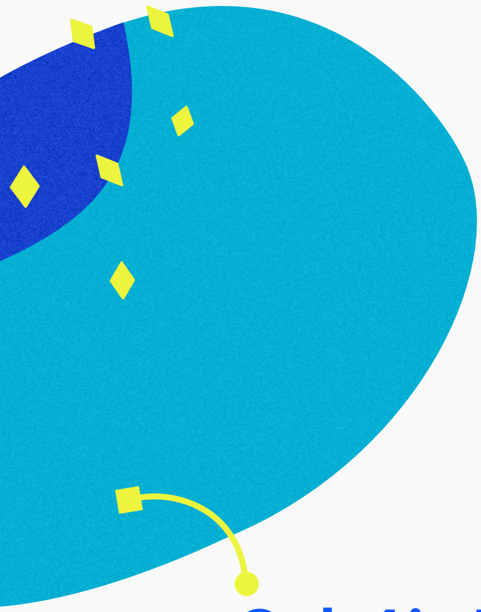
Quality testing

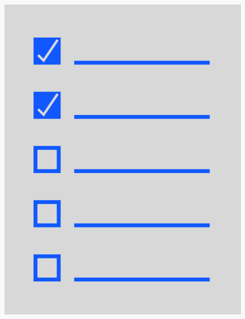
Robust quality testing is paramount for establishing true software quality and resilience. Ensuring that code is thoroughly validated across environments, from unit tests to functional automation, is critical for catching defects early and preventing issues from reaching production. Without comprehensive testing, organizations risk compromising reliability and eroding user trust, directly impacting their ability to deliver high-quality, resilient systems.

Despite its importance, there are notable gaps in quality testing practices:

60%

of engineering leaders say their test or staging environments do not match the production environment.



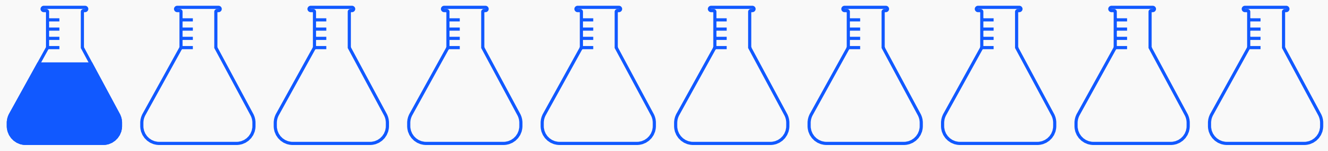


A quarter (25%) of engineering leaders say less than

30%

of their features have test plans.

Only 1 in 5 (20%) engineering teams have more than 90% unit test coverage for their services.



Just 10% of engineering leaders say functional testing is fully automated.

To mature quality testing, prioritize achieving high test and production environment parity. Invest in automating functional and unit tests, define clear test plans for all features, and integrate continuous verification into your deployment pipelines to ensure automated quality gates and faster feedback loops.

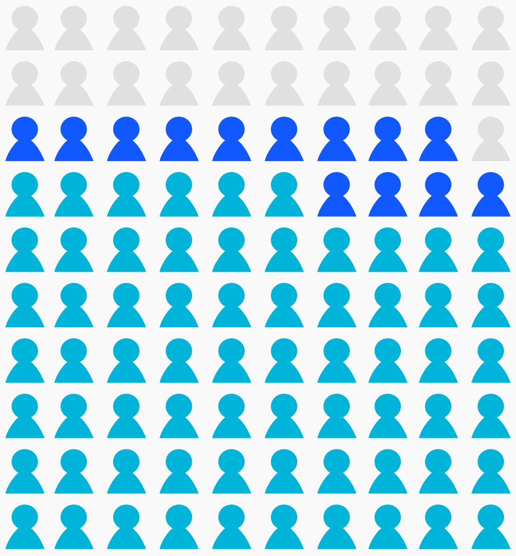
Resilience testing

Modern engineering practices integrate chaos testing into their DevOps lifecycle for a truly resilient and high-quality system. This advanced practice allows teams to stress test their software in extreme conditions, such as seasonal spikes or emergency scenarios, ensuring it can handle unusual demand and maintain stability when unforeseen events occur. This proactive approach is fundamental to building robust, fault-tolerant systems.

However, the adoption of chaos testing remains limited:

66% of engineering teams do not use chaos testing, while only

13% have integrated it into the SDLC.

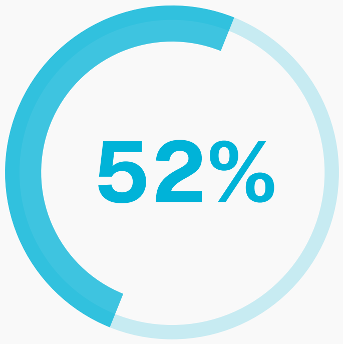


Integrate dedicated chaos testing practices and tools into your SDLC to mature resilience testing. Proactively simulate failures and extreme conditions in controlled environments to identify weaknesses, validate system behavior under stress, and ensure your applications can gracefully recover from unexpected events.

Incident management

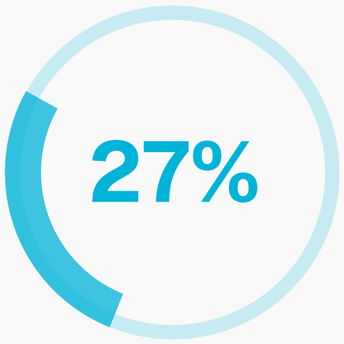
However mature their practices, experience has proven that engineering teams must always be prepared for the unexpected. If something can go wrong, then it will. As such, it's essential to equip them with the tools and processes to resolve incidents quickly and learn from them. This preparedness is fundamental to maintaining system quality and resilience, minimizing downtime, and fostering continuous improvement.

Despite this necessity, gaps persist in incident management:



More than half of engineering teams **do not have key tools** to support incident management.

1 in 10 engineering teams **don't have a formal incident management process**.



Only a quarter of engineering leaders say blameless **post-mortems are conducted** for all of their incidents.

To mature incident management, establish a formal process with clear roles and responsibilities, supported by integrated tools for rapid detection, communication, and resolution. Crucially, cultivate a culture of blameless post-mortems to learn from every incident and systematically enhance system resilience.

The Bottom Line

Prioritizing software quality and operational resilience is key to minimizing disruptions and maintaining business continuity. Proactive testing, resilient systems, and strong incident management significantly reduce outage impact. For every 1000 developers, organizations have an opportunity to realize the following savings:

Formula:

Critical Outages	=	3 outages per year x 6 hrs per incident	x	$200k	per hour from revenue loss & penalties	=	$3.6M	loss per year
Minor Outages	=	12 outages per year x 2 hrs per incident	x	$75k	per hour from productivity loss & overtime wages	=	$1.8M	loss per year

Total annual downtime cost = **$5.4M**	**40 to 60%** reduction from quality & resiliency measures	**$2M - $3M** annual savings

05: Secure Software Development

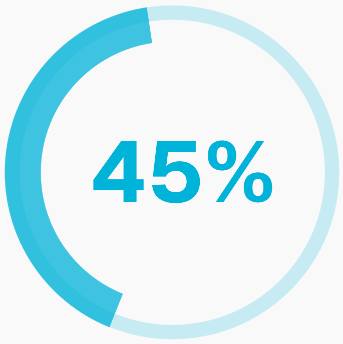
Establishing a truly mature engineering practice requires security to be an intrinsic part of every development phase, not an afterthought. Secure software development involves integrating robust security and governance policies early, maintaining comprehensive visibility into software components through tools like SBOMs, and ensuring developers are trained in best practices. This holistic approach builds resilience against threats, minimizes vulnerabilities, and safeguards critical assets throughout the software lifecycle.

Integrated security and governance

Mature engineering practices recognize that shift-left is not about pushing additional work onto developers, but empowering them to build secure software by default. To succeed in secure software development, engineering leaders must reduce the toil of secure software delivery by integrating security and governance seamlessly into every lifecycle stage. This proactive approach ensures compliance and minimizes vulnerabilities from inception to deployment.

However, organizations often face significant gaps in this critical area:

of engineering leaders say that most (more than two-thirds) of their build **pipelines are not gated with security scans**.



of engineering leaders say it takes **more than 7 days on average to resolve** high-severity security issues.

Nearly 1 in 10 engineering leaders say **high and critical severity bugs are not fixed** prior to release.

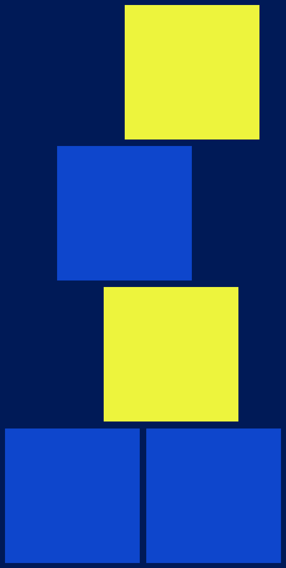
To mature integrated security and governance, establish clear security best practices. Automate security scanning and policy enforcement within the CI/CD pipeline, and ensure that high-severity issues are identified, prioritized, and remediated swiftly before release.

Software Bill of Materials (SBOMs)

Understanding the precise composition of your software is no longer optional for robust, secure software development and a mature engineering practice. A Software Bill of Materials (SBOM) provides critical transparency into every application component, dependency, and vulnerability. This granular visibility is essential for proactive risk management and building more secure systems.

However, the adoption of SBOM generation is still limited:

Only 11% of engineering leaders say a software bill of materials (SBOM) is generated for all artifacts



Over half of engineering teams (57%) generate SBOMs for

less than 30%

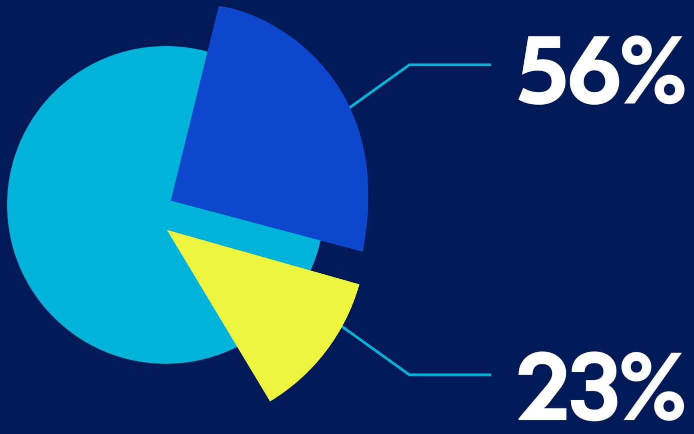
of artifacts, including not at all.

To mature your SBOM practice, establish processes for automated SBOM generation for all software artifacts across the development lifecycle. Integrate SBOMs with vulnerability management tools to enhance risk identification and ensure compliance with evolving industry standards and regulatory mandates.

Security training

For a truly mature engineering practice, security training for developers is indispensable. Even with advanced tools and automated checks, human awareness of security best practices remains a critical line of defense. Equipping engineers with up-to-date knowledge on secure coding, common vulnerabilities, and threat landscapes empowers them to build secure software from the ground up, reducing risks and improving overall software quality.

However, the consistency of such training varies:



While over half of developers **(56%) are trained annually or semi-annually on security best practices, nearly a quarter (23%) are never trained.**

To mature security training, establish a continuous and formalized curriculum for developers covering current security best practices, secure coding principles, and common vulnerabilities. Regular training, workshops, and access to security resources can significantly elevate the security posture of your development teams.

The Bottom Line

Integrating security early and automating scans reduces developer toil. While security toil is just the tip of the iceberg for integrated security, its reduction offers significant productivity gains. This calculation focuses solely on these productivity gains, not the vast savings possible from preventing breaches or brand damage. For every 1000 developers, organizations have an opportunity to realize the following productivity savings:

Formula:

1k Devs x $100k Annual Salary = **$100M**

1k Devs x 2000 hours = **2M** annual work hours

5% Time per dev per year on manual security tasks & vulnerability remediation = 100k hrs /year x $50 per hour = **$5M** per year

20-25% Reduction from automated scans and remediation

$1M - $1.25M annual savings (just in productivity)

Your Next Steps

Today’s engineering teams carry a significant burden of responsibility as they strive to balance rapid innovation alongside improved software security and resilience. Mature engineering practices, underpinned by end-to-end pipeline automation, are essential to their ability to build, test, and deploy their software faster and with greater confidence.

However, the findings of this report highlight that many organizations remain in the early stages of their journey to engineering excellence across key dimensions. While there is a widespread adoption of best practices, adherence to them is often inconsistent, used in only isolated use cases or pipeline stages. This fragmented approach leads to inefficiencies and a lack of consistency in how organizations approach Developer Experience, DevOps Modernization, Optimization, Quality & Resilience, and Secure Software Development.

To accelerate their journey toward engineering excellence, leaders must move beyond traditional, siloed approaches and embrace a more unified approach to software delivery. Adopting a platform-centric strategy that seamlessly integrates the tools and capabilities developers need across these critical areas will set engineering leaders on the surest path to success.

We invite you to take the Engineering Excellence Assessment to assess your organization's progress in these vital aspects of engineering maturity and more.